IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application Ser. No.: 09/466,627 Group Art Unit: 2176

Filing Date: 12/17/1999 Examiner: M. NGUYEN

Attorney Docket Number YO999-429 Inventor Name(s): LO ET AL.

Title: METHOD AND APPARATUS FOR CONVERTING BETWEEN DATA SETS AND XML DOCUMENTS

Commissioner for Patents
P.O. Box 1450
Alexandria VA 223131-1450

DECLARATION OF SHYH-KWEI CHEN, PH.D.

Sir:

I, Shyh-kwei Chen, Ph.D., hereby declare as follows:

1. I am one of the named inventors in the above-identified application. My co-inventor, Ming-ling Lo, is no longer employed by the assignee of this invention, IBM. Accordingly, he is not readily available and is not joining me on this declaration. Nevertheless, I believe that he would agree with what I declare here.

2. Attached as Exhibt A is a copy of a presentation that I made with my co-inventor Ming-ling Lo and my manager Jen-Yao Chung in December of 1998 discussing the project that Ming-ling Lo and I did relating to XML. This exhibit shows, on page A-3, that we planned to work full time on the project during the entire year of 1999.

3. Attached as Exhibit B is an invention disclosure that I created on April 19, 1999, with my co-inventor Ming-ling Lo, and which matured into the above-identified patent application. I have been informed and believe that this exhibit is a copy of a business

record maintained by the IP Law Department of my employer and assignee, IBM corporation. I therefore trust its accuracy. .

3. Enclosed as Exhibit C is a printout of a computer directory relating to files and/or e-mails relating to this project. These files and/or e-mails have system-generated creation dates that show continuous work on the project that matured into the above-identified patent application from June of 1999 through October of 1999. These documents have titles such as "DTDSA" and "XML," which I recognize as pertaining to this project. These system-generated creation dates are business records that are maintained by the system software of my laptop computer. I do not know how these dates could be altered. I therefore trust their accuracy.

4. Based on looking at Exhibit C, my recollection is refreshed and I also remember that there were earlier files and e-mails dated continuously, throughout the first half of 1999, and relating to this project. These e-mails were destroyed when I got a new laptop on or about June of 1999.

5. Enclosed as exhibit D is a directory printout from my laptop showing the system creation date of exhibit A, namely 12/15/1998.

6. Based on the documents identified above, my recollection is refreshed regarding the events of 1998 and 1999. I therefore remember that Ming-ling Lo and I

   - conceived of the idea of establishing a mapping from lists and scalars corresponding to at least one data source into XML elements and attributes during the summer of 1998;
   - worked at least part time during the fall of 1998 to reduce this invention to practice; and

- worked full time reducing this invention to practice during the entire year of 1999.

7. I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

[illegible]

Shyh-kwei Chen, Ph.D.

6/9/2004

Date

A-1

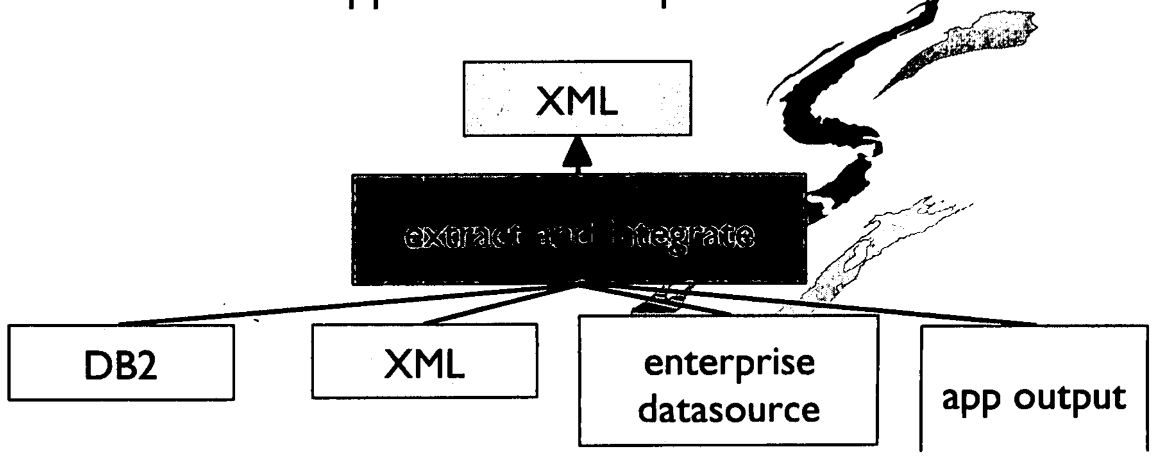
XML Access Server

IBM T.J.Watson Research Center
Ming-Ling Lo
Shyh-Kwei Chen
Jen-Yao Chung

Exhibit A

XAS Project Goals

- Machine generated virtual XML documents
- Dynamically extract data from distributed, heterogeneous enterprise data sources, deliver as XML documents
  - Data stored in existing data sources
  - XML as data presentation and transmission format
  - Materialize into XML documents when necessarily
- Motivations: Large amount of data already exists in data systems such as RDB
  - Maximizing investment: re-use data and schema design
  - Avoid re-inventing wheels: business logic already written for RDB, etc; co-exist with existing applications
  - Better data integrity and quality control: one copy of data
  - Connecting to new world: build new eCommerce application on top of XML interface

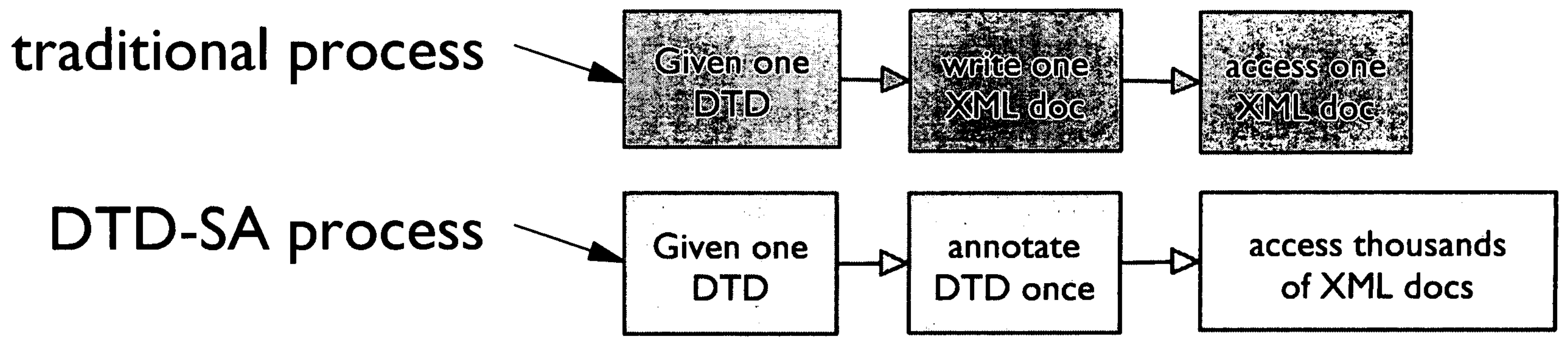


Project Content

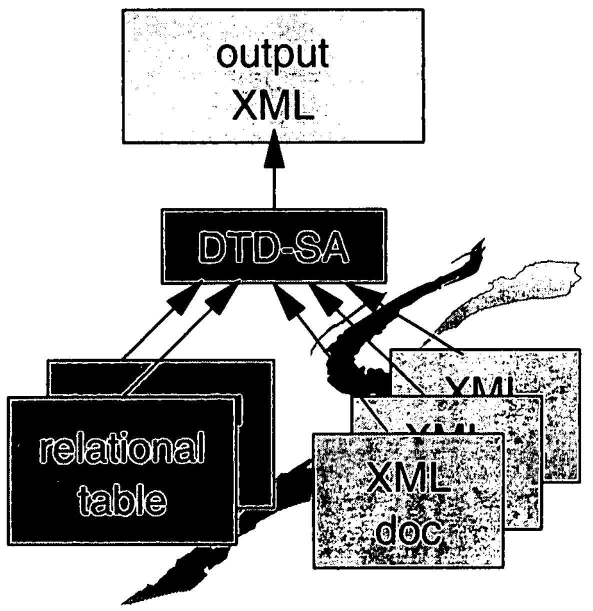
- Underlying technology:
  - Approach: Once per type authoring - one authoring effort per DTD, enable access to hundreds/thousands of XML documents of this DTD
  - Rigorous framework for DTD to existing schema mapping
  - Algorithms for type-by-type authoring
    - Enterprise data publishing: given existing data, publish as XML
    - Schema layout given DTD: find best table layout for DTD
    - Cross schema matching: Given DTD and RDB schema for similar purpose, establish mapping
    - Componentized XML construction: Use existing data as building blocks of new XML docs
- Server architecture design and prototyping
  - XML document retrieval, deposit, and query
- Timeframe: 1-12/1999

A-4

Mapping Technology: DTD-SA

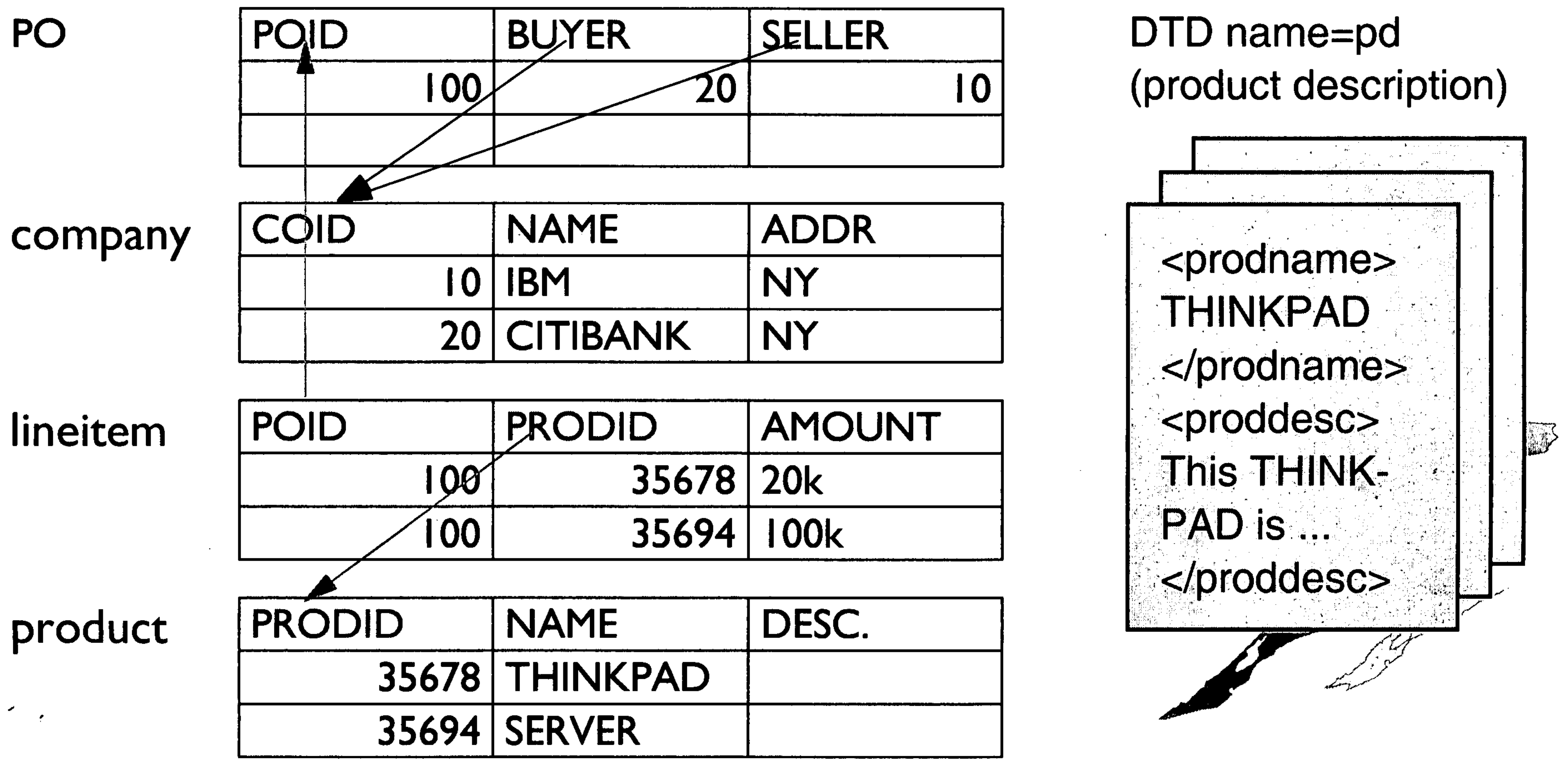


- DTD-SA = Document Type Definition with Source Annotation
- Framework for specify mapping between DTD and existing schema
  - Existing schema
    - relational-like or DTD-like
- Essentially looks like DTD, but with annotations:
  - value specifications
  - binding specifications
  - Simply DTD when stripped of annotations



Example - Purchase Order

- Purchase order related information described by 4 tables and 1 DTD



PO

POID	BUYER	SELLER
100	20	10

company

COID	NAME	ADDR
10	IBM	NY
20	CITIBANK	NY

lineitem

POID	PRODID	AMOUNT
100	35678	20k
100	35694	100k

product

PRODID	NAME	DESC.
35678	THINKPAD	
35694	SERVER	

DTD name=pd
(product description)

```
<prodname>
THINKPAD
</prodname>
<proddesc>
This THINK-
PAD is ...
</proddesc>
```

Example (cont-1)

- A purchase order DTD

```
<!ELEMENT PO (id, buyer, seller,  (lineitem)* )>
<!ELEMENT id (#PCDATA)>
<!ELEMENT buyer (name, address)>
<!ELEMENT seller (name, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT lineitem (prodname,   proddesc, amount)>
<!ELEMENT prodname (#PCDATA )>
<!ELEMENT proddesc (#PCDATA )>
<!ELEMENT amount (#PCDATA )>
```

Example (cont-2)

- Add annotation
  - blue: value specification
  - red: binding specification

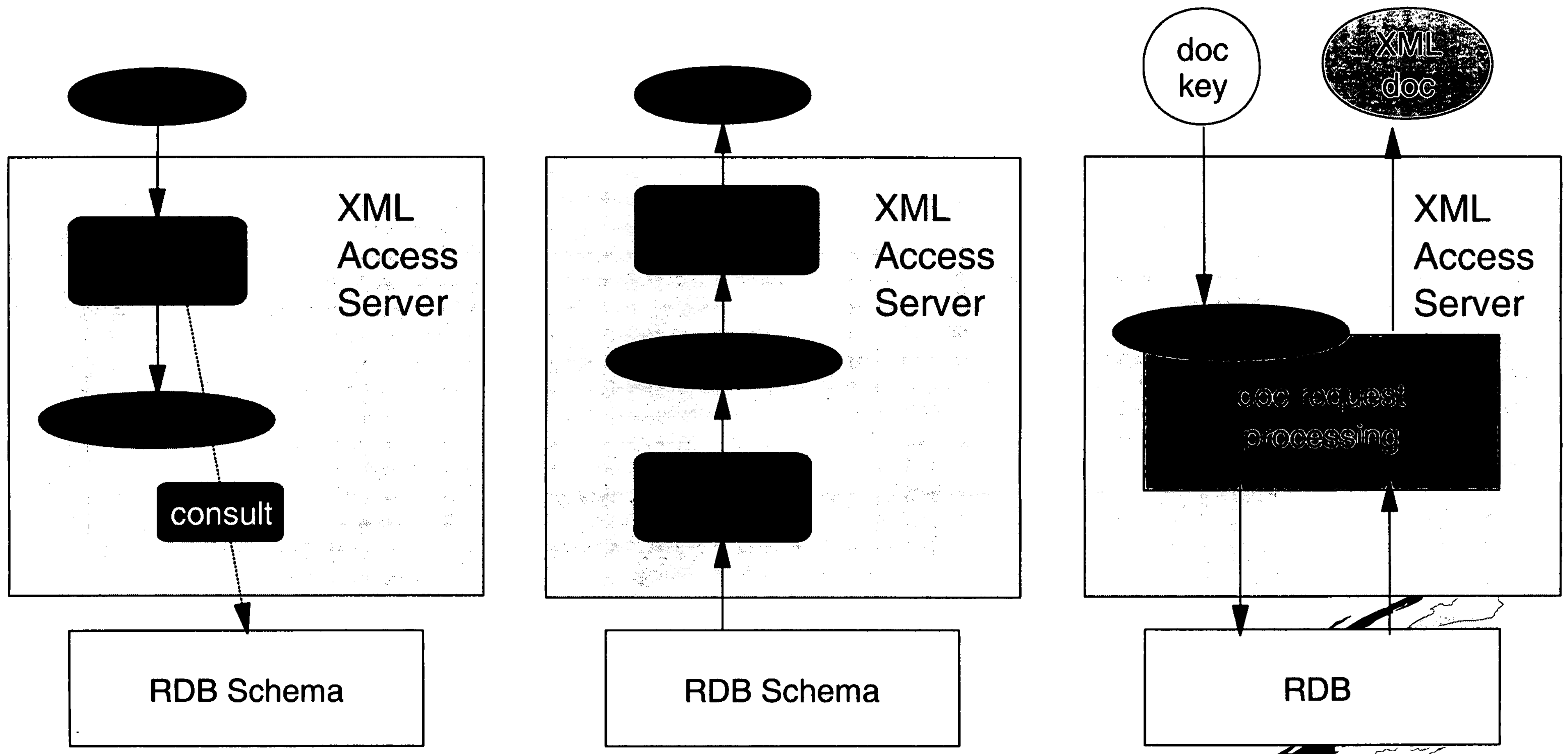
```
<!ELEMENT PO (id, buyer, seller,  (lineitem)* :: w:= row(lineitem, poid, PO.poid(r))   )>
                :: r:=row(PO, poid, x)
<!ELEMENT id (#PCDATA :PO.<POID>(r) )>
<!ELEMENT buyer (name, address)> :: s:= row(company, id, PO.buyer(r))
<!ELEMENT seller (name, address)> :: s:= row(company, id, PO.seller(r))
<!ELEMENT name (#PCDATA :company.name(s) )>
<!ELEMENT address (#PCDATA :company.addr(s) )>
<!ELEMENT lineitem (
                prodname,
                proddesc :: d:=doc( pd, root.description, product.prodname(v)),
                amount)> :: v:= row( prod, prodid, lineitem.prodid(w))
<!ELEMENT prodname (#PCDATA :product.prodname(v) )>
<!ELEMENT proddesc (#PCDATA :pd.description(d) )>
<!ELEMENT amount (#PCDATA :lineitem.amount(w) )>
```

Example (cont-3)

- Retrieve PO document with PO-ID=100 (x=100)

```
<PO>
    <id> 100 </id>
    <buyer>
        <name> CITIBANK </name> <address> NY </address>
    </buyer>
    <seller>
        <name> IBM </name> <address> NY </address>
    </seller>
    <lineitem>
        <prodname> THINKPAD </prodname>
        <proddesc> This THINKPAD is quite good </proddesc>
        <amount> 20K </amount>
    </lineitem>
    <lineitem>
        <prodname> SERVER </prodname>
        <proddesc> This server is the best </proddesc>
        <amount> 100K </amount>
    </lineitem>
</PO>
```

DTD-SA Deployment



1. Annotate given DTD to generate DTD-SA

2. Or construct DTD-SA based on existing RDB schema

3. Use DTD-SA to guide XML doc access

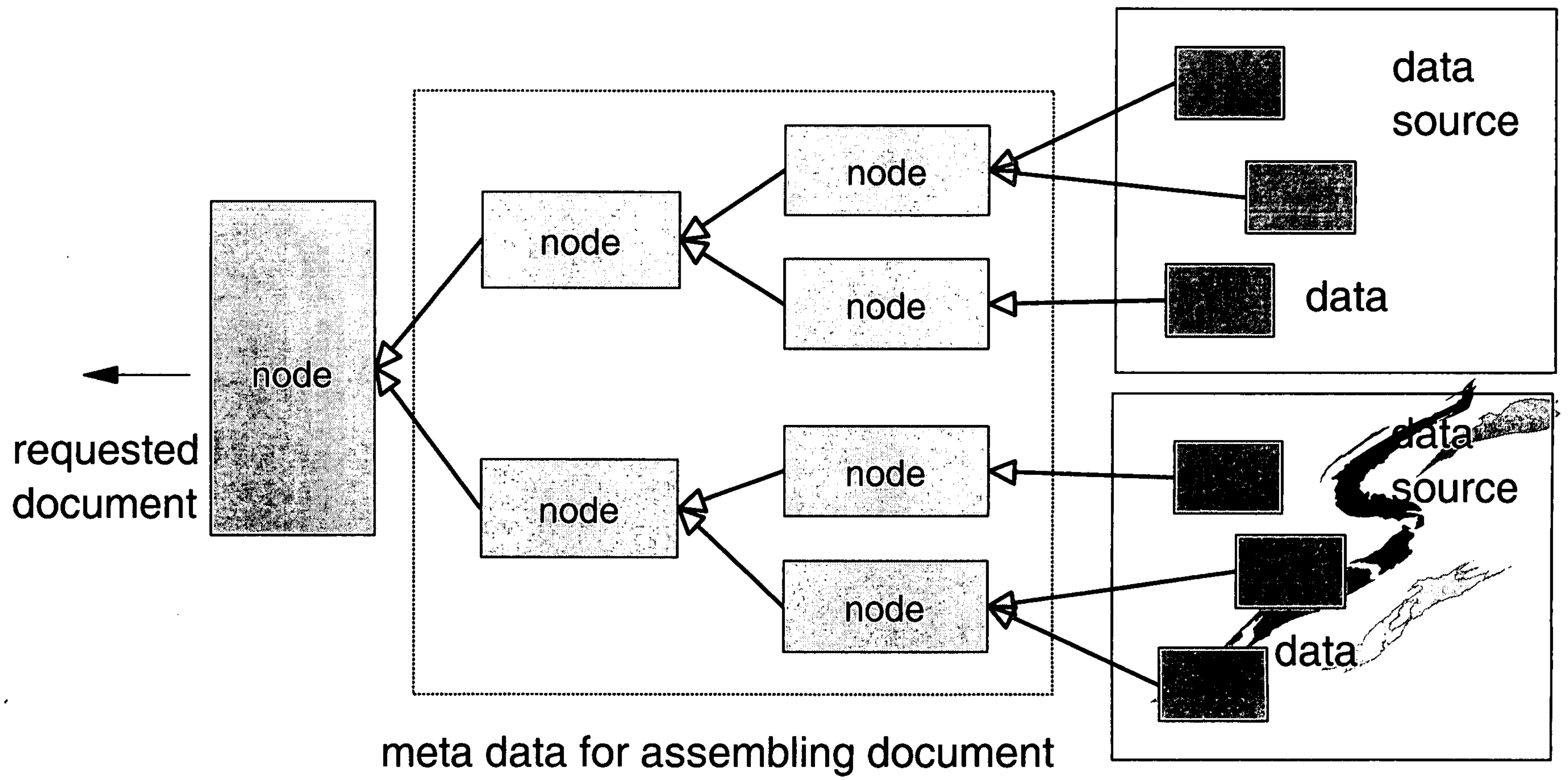
Advantages

- One target DTD can be composed from multiple
  source tables, schemas, systems (source docs, DTD,repositories)
- Seamless integration of raw data and XML data
  - An output XML document can be woven from mixture of
    - relational columns with raw format data
    - relational columns with XML format data
    - contents of other XML documents
- No changes to source system, schema, and data
- One source schema can map to infinite number of DTDs
- Root element target DTD can start from
  any source table in a network of the foreign key relationships
  (any documents in a collection of source XML documents)
- Mathematically rigorous: foundation for automatic algorithms

A-11

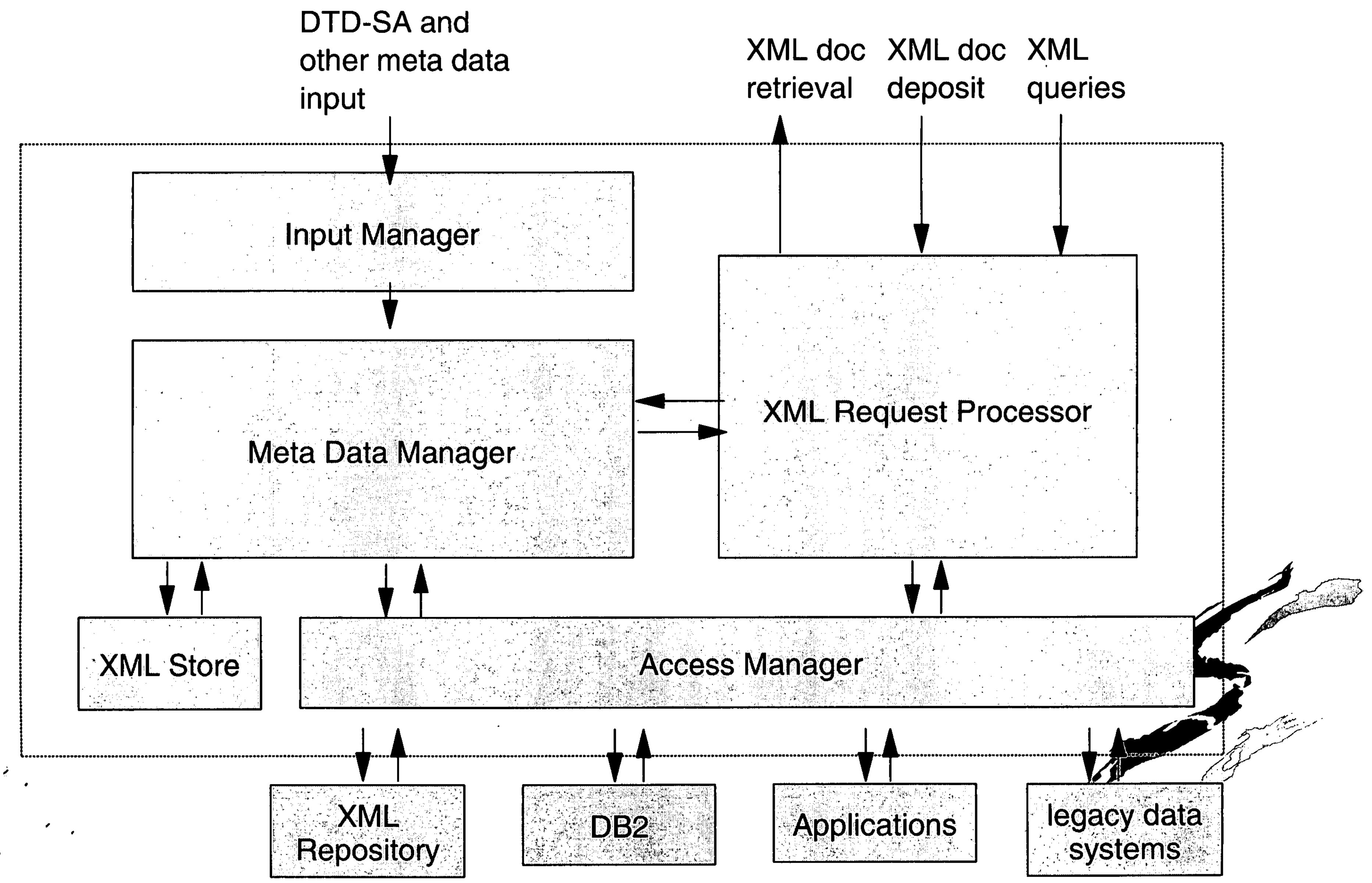
Document Access Implementation

- Doc access translated into multiple enterprise DS accesses
- Assemble result from lower level results.
- Manage meta data for assembling XML doc



A-12

XAS Overall Architecture



Project Progress

- Accomplished
  - Developed a new technology for mappings between relational schema and XML DTD, called DTDSA
  - Designed architecture framework
  - Ability to extract data from DB2 to XML using DTDSA
- Currently under work
  - Testing XML extraction using DTDSA on 390 environment
  - Storing XML to single source using DTDSA
  - Investigation 390 specific data source, such as VSAM
- Next steps
  - XML as data sources
  - Multiple data sources
  - Developing an algorithm for constructing DTD-SA
  - Scalable meta data management: DCL Graphs
  - Query XML using DTDSA as underlying mechanism

OIPE JC151 JUN 1 7 2004 PATENT & TRADEMARK OFFICE

Exhibit B

Disclosure YOR8-1999-0350

Created By: Ming-Ling Lo **Created On:** 04/19/99 05:03:44 PM
Last Modified By: Ming-Ling Lo **Last Modified On:** 04/21/99 12:01:33 PM

*** IBM Confidential ***

Required fields are marked with the asterisk (*) and must be filled in to complete the form .

Summary

Status	Submitted
Processing Location	YOR
Functional Area	900 Systems & Software-Goyal
Attorney/Patent Professional	Kevin M Jordan/Watson/IBM
Submitted Date	04/21/99 12:00:03 PM
Owning Division	RES
PVT Score	

Inventors with Lotus Notes ID's

Inventors: Ming-Ling Lo/Watson/IBM, ShyhKwei Chen/Watson/IBM

Inventor Name > denotes primary contact	Inventor Serial	Div/Dept	Manager Serial	Manager Name
> Lo, Ming-Ling	013988	22/9AUD	653458	Padmanabhan, Sriram
Chen, Shyh-Kwei	707253	22/9AUF	520894	Chung, Jen-Yao (Chung)

Inventors without Lotus Notes ID's

IDT Selection

IDT Team:	Attorney/Patent Professional: Kevin M Jordan/Watson/IBM

Main Idea

***Title of disclosure (in English)**

A framework for mappings between XML DTD and relational databases

***Idea of disclosure**

1. Describe your invention, stating the problem solved (if appropriate), and indicating the advantages of using the invention.

Problem Description:

XML is emerging as one of the most important format for document and data representation and transmission. Many users and new applications require their input and output to be in XML format. For XML documents, there is the concept of a Document Type Definition (DTD). Each DTD describe the structure of a (potentially infinitely large) set of XML documents. An XML document either has no associated DTD or belongs to exactly one DTD. When an XML document belongs to a DTD, its structure must conform to the specification of the DTD.

There is a large quantity of data already exist in relational databases. These databases are

B-1

designed without the expectation that they may one day be accessed for presenting the access result as XML documents. These databases may also have existing applications running on them, still depending on accessing the data in their original format. Furthermore, the data stored in such databases may lack XML specific information such as element tag names or attribute names. Given these facts, it is nonetheless very derivable to access those data in XML format.

High level description of invention:
This invention discloses a framework for creating mappings between XML DTDs and relational database schemas. The relational database can be regarded as having many virtual XML documents of many DTDs stored in them. Each mapping created by our framework specified a subset of these XML documents (which belong to one DTD) explicitly.
Given a DTD to the relational database mapping, when a request for a document of this DTD arrives, the request is automatically translated into accesses to various data items in the database. The accessed data items are then assemble to form the requested XML document. The mapping directs which data items to access, and into which element the accessed data item should go.

Advantages:
The advantages of this approach are:
1. Given such a mapping, those who want to access XML document from relational databases no longer need to be concerned with the details of relational databases. All access and query can be expressed in XML terms.
2. Data in relational database need not be explicitly converted into XML format, saving time and space required for such conversion. Also, because only one copy of data is kept, there is no consistency problem between multiple copies of the same data to worry about.
3. It is not necessary to change the relational database system, the relational schema, or the data itself for the need of exporting relational data in XML format. Existing relational database applications can continue to run without modification.
4. The framework supports flexible mappings. In particular, each relational table is not limited to just one "natural" of default mapping. Each table may as many DTDs defined 1. on them as necessary. Also, a DTD is not limited to map to only one relational table. Instead, each DTD can map to multiple relational tables. The foreign key relationship is incorporated into the framework, and expressed seamlessly in the framework.
5. The framework is mathematically rigorous, and can thus be more efficiently and reliably be implemented.
6. The application of this framework is not limited to relational databases. Any data source that can be modeled in a relational manner (such as those expressible by ODBC or JDBC standards) can be used as a data source in our framework.

2. How does the invention solve the problem or achieve an advantage,(a description of "the invention", including figures inline as appropriate)?

Framework for Mappings

This invention defines a set of syntax and binding rules for mappings between XML DTD and relational databases, call DTD-SA (Document Type Definition data Source Annotation)

Notation:

Symbol	Denoting
T	a relational table
C	a column C; assuming each column identifier is unique with the schema
T.C	same as C, but emphasizing the fact that C is a column of table T
Cx, Cy, Cz	Column variables, whose values are column identities
K	a column value
Kx, Ky, Kz	column value variables
<C>	array of columns <C1, C2, ..., Cn>
<K>	an array of column values <K1, K2, ...Kn>. Note: K1, K2, ...Kn may be in different domains.
<<K>>	a sequence of <K>
row(<C>, <K>)	a function defined only when C1, C2, ... Cn belongs to the same table, and Ki is in the domain of Ci, for all i=1,2...,n. The output of this function is a sequence of rows in table T, with C1=K1, C2=K2, ...Cn=Kn.
T.<C>()	a function which takes a sequence of rows and returns the projection of the sequence in columns <C>
T.C()	shorthand of T.<C>() when there is only one column in <C>

The composite function T.<C1>(row(<C2>, <K>)) can be abbreviated as T.<C1>(<C2>=<K>), or as T.<C1>(<K>) when the identity of <C2> is obvious (e.g. primary key).

Basic Terminology

Specification time: the time when a DTD-SA is specified.

Runtime: the time when a document is either retrieved or queried against.

Basic syntax of DTD-SA

In the following discussion, the function family F<Cout>(r, n), which is a cleaner form for f(T.<Cout>(r), n), where <Cout> is a vector of column identify constants, r is a row variable, and n is an integer, appears many times. The syntax has the following meanings:

1. r is unbounded at specification time and bounded only at runtime.
2. All columns in <Cout> must belong to the same table, T (as is evident from the

B-3

syntax T.<Cout>).

The sub-construct r := row(<C>, <K>) also appears many times. The output of row(<C>, <K>) is a sequence of rows. And the semantics of the construct is:
3. The rows in the sequence are bound to row variable R in turn.

DTD-SA specification rules
The following table lists the original DTD constructs and their annotated counterparts in DTD-SA. For each DTD construct listed a row whose requirement is "must", the construct must be written as the DTD-SA construct as described in the 2nd column. For each DTD construct list in a row whose requirement is "may", it may be replaced with DTD-SA construct listed in the 2nd column, but is not required to. All other DTD construct remains the same in DTD-SA format.

DTD construct	DTD-SA construct	requirement
#PCDATA	#PCDATA :F<Cout>(r, n)	must
#CDATA	#CDATA :F<Cout>(r, n)	must
X, for X=NMTOKEN, NMTOKENS, ID, IDREF, ENTITY, ENTITIES, NOTATION, Enumerated NOTATION	X:F<Cout>(r, n)	must
...*	...* ::n := F<Cout>(r, m)	must
...*	...* ::r := row(<C>, <K>)	must
...?	...? :: n:=F<Cout>(r, m)	must
(x1x2...xn)	(x1x2...xn) :: n:=F<Cout>(r, m)	must
<!ELEMENT E (....)>	<!ELEMENT E (...)> ::r := row(<C>,<K>)	may

Some terminology can be defined based on the above DTD-SA constructs.
Each DTD-SA construct listed above has either a *content specification* or a *binding specification* associated with it. The content specification is marked by a leading ":", while the binding specification is marked by a leading "::". An content or binding specification is sometimes called a *spec* for convenience. In a DTD-SA, an element definition that contains or is appended with content or binding specifications is called an *annotated element definition (AED)*. Similarly an attribute definition with content or

binding specifications is called an *annotated attribute definition (AAD).* An AED or AAD is sometime called simply an *annotated definition (AD)* for convenience.

Basic Semantics of DTD-SA Construct

A DTD-SA specification consists of a list of DTD specifications, which when stripped of the content and binding specifications, is simply a DTD.

At the high level, DTD-SA works in the following way: during document retrieval time, a set of row variable bindings is supplied to the DTD-SA, causing all unbounded variables in the DTD-SA to become bounded. In the process, the value, or content, for each element and attribute will be generated. The content of the whole XML document will be generated as a result.

The content specification may have unbounded variables. A content specification denotes that, during document retrieval, the content of its associated element is the output of the function described in the content specification, with all variables, if any, bound to some runtime supplied or derived values.

The binding specification specifies that some unbound row variable in its descendent elements and attributes are bound to the value or expressions listed in the specification. The binding specification generally has the form r := row(<C>, <K>), which means the row variable r in the descendent elements will be bound to the output of row(<C>, <K>). The output of row(<C>, <K>) is a sequence of rows. If the DTD construct being annotated is a regular element definition, the first output row is bound to the element. If the DTD construct being annotated is a "*" construct, r inside the "*" construct and in all its descendent elements will be bound to the output of row(<C>, <K>) in turn. The values generated using all these bindings will be the content of the "*" construct.

The specific semantics of the above constructs are further explained below:

a. The integer parameter n:

e.g. #PCDATA :f(T.<C>(r), n)

The use of parameter 'n' is usually for expressing the number of times the annotated #PCDATA appears in its enclosing construct. The parameter n in f(T.C(), n) is especially useful when #PCDATA appear inside the * construct.

Note:

i. In most cases f() is actually an identity function.

ii. In the cases f() does not really depend on n, we will shorthand the function as f(T.<C>(r)). Likewise we can shorthand the function as f(n) or f() when appropriate.

b. ...* :f(T.C(r), n):

in this construct, the output domain of f() is non-negative integers. The output integer m determines the number of repetition for the "*" construct. The numbers 1, 2, ...m will be bound the unbound variable n insides the "*" construct in turn. The result of these bindings will be the content of the "*" construct.

c. ...* ==> ...* ::r := row(<C>, <K>)

Bind r to the output row(<C>, <K>) in turn. The number of repetition of the * construct is the number of rows in the output.

d. ...? ---> ...? :f(T.C(r), n)

The output domain of f() is {0,1}. The output value determine the whether the construct inside the "?" construct occurs once or does not occur.

e. (x1x2...xn) ==> (x1x2...xn) :f(T.C(r), n)

The output domain of f() is {1,2,...,n. The output value determines which of the alternatives to take.

f. <!ELEMENT E (....)> ==> <!ELEMENT E (...)> ::r := row(<C>,<K>)

binding the row variable r to the output of row(<C>, <K>). If row(<C>, <K>) output more than one rows, r is bound to them in turn. And the content of the construct is a sequence of elements E

Binding rule:

Variables of the same name may appear in various places in a DTD-SA. They may or may not bind to the same value. It is therefore necessary to define the binding rules clearly. There are two types of unbound variables in a DTD-SA, row variables, and integer variables used in determining repetition and alternatives selection.

Distance rules for content and binding specifications:

1. For two specs s1 and s2 in the same annotated definition, s1 is an ancestor spec of s2 if s2 is followed, among other symbols, by some number of ")" then by s1, in the annotated definition.

2. If s1 and s2 are both ancestor specs of s3 in the same annotated definition, s1 is closer to s3 then s2, if the number of ")" between s1 and s3 is less.

3. If s1 is defined in annotated definition e1, and s2 defined is annotated definition e2. s2 is an ancestor spec of s1 if e2 is an ancestor annotated definition of e1.

4. An ancestor spec in the same annotated definition is always closer to an ancestor spec in an ancestor annotated definition.

5. If e1 and e2 are ancestor annotated definitions of definition e3, and s1, s2 and s3 are defined inside e1, e2, and e3 respectively. s1 is closer to s3 then s2, if e1 is closer to e3 then e2.

Binding rules for row variables

6. A row variable r binds to the closest ancestral row binding specification with the same left-hand side (i.e. :: r:= ...).

Binding rules for integer variable

7. An integer variable n binds to the closest ancestor integer binding specification with the same left-hand side (i.e. :: n:= ...)

How to Access an XML Document using DTD-SA?

To retrieve an XML document using a DTD-SA, one or more row variable bindings must be supplied to the root element, so that all unbound variables in the descent elements and

attributes are bound.
XML retrieval
To access an XML document,
1. (necessary only is DTD-SA system, not needed for single DTD-SA)
Select an element as root element.
2. Supply a "r:= row(<C>, <K>)" to the root element.
Everything else will follow through naturally.

Mixing XML data and raw data:
One great advantage of this method is that the data retrieved from the database can be in either raw format or XML format. The above discussion already explains how data in raw format can be retrieved and put into XML format.
A field in a database table can have either raw (non-XML) data or XML data. Our framework handles both case smoothly. To incorporate data field with XML formatted data, we introduce the two addition DTD-SA specification rules:

DTD construct	DTD-SA construct	requirement
<!ELEMENT E (....)>	<!ELEMENT E (...)> :F<Cout>(r, n)	may
<!ATTLIST E A (....)>	<!ATTLIST E A(...)> :F<Cout>(r, n)	may

The rule for element E means the content of the element E will be replaced by the output of F(), which may contain XML tags. The content of the output of F() is subject to the usual XML parsing and syntax checks. Similar semantics applies to the rule for attribute A.

3. If the same advantage or problem has been identified by others (inside/outside IBM), how have those others solved it and does your solution differ and why is it better?

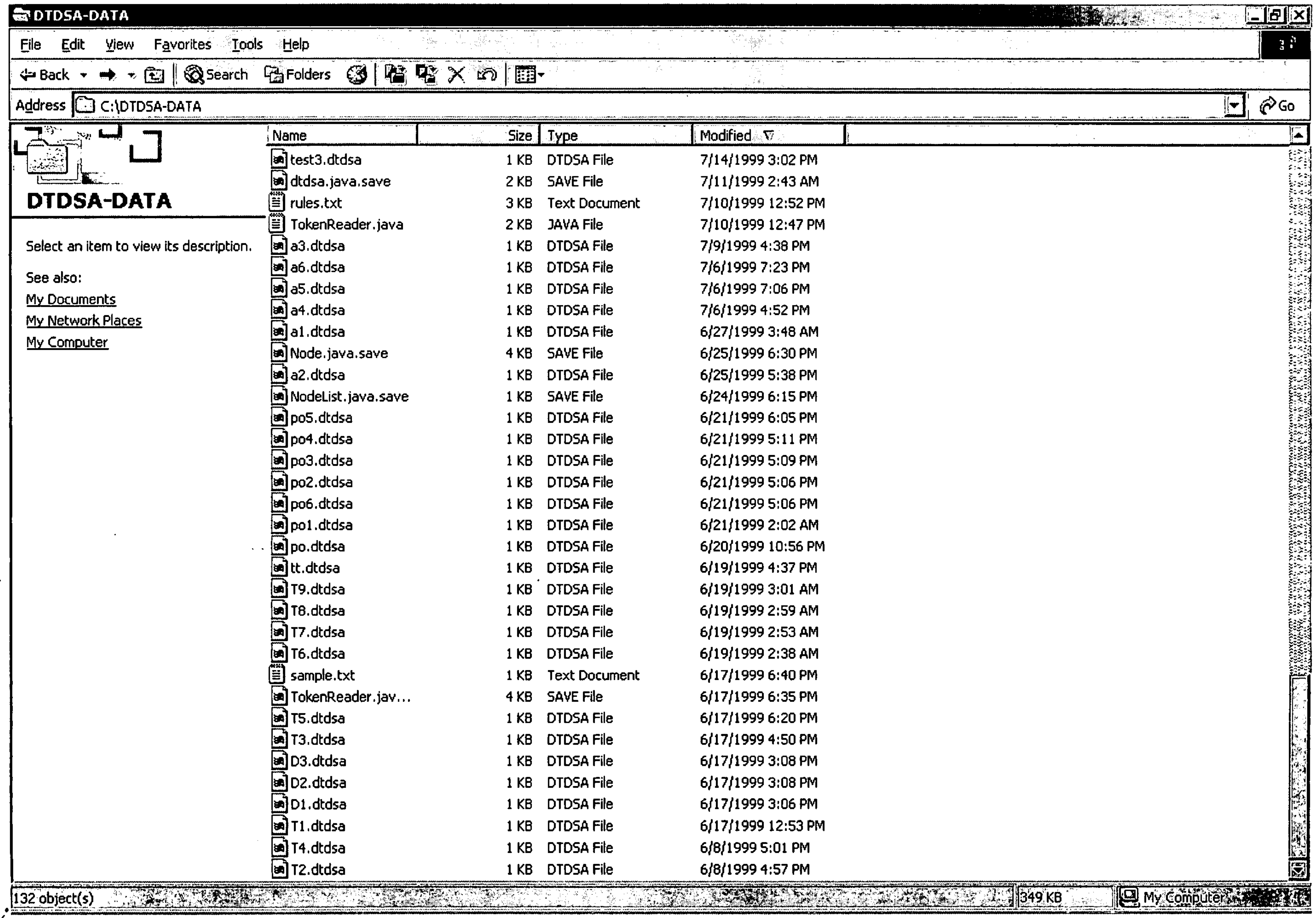
Comparison with related approaches
There have not been prior inventions related to mappings between XML DTD and relational schema. We therefore compare our invention with the following intuitively conceivable approaches.

Without such a framework for creating mapping between XML DTD and relational schema, there can be the following approaches:
1. For each relational table, assign a default DTD based on the schema of the table. Such an approach is too inflexible. If the DTD is not exactly what the application need, more work is require to convert the DTD to the desired one. Also a DTD is limited to a single table.
2. Issue SQL command, and translate the result to XML format. Again the result format is not flexible enough and has only limited usefulness.
3. Write a potentially complicated SQL program to access the relational database, then fill

the elements of the requested document with the data. The SQL can be complicated, and the logic for filling XML document elements is ad hoc and hardwired in the program.

Exhibit C

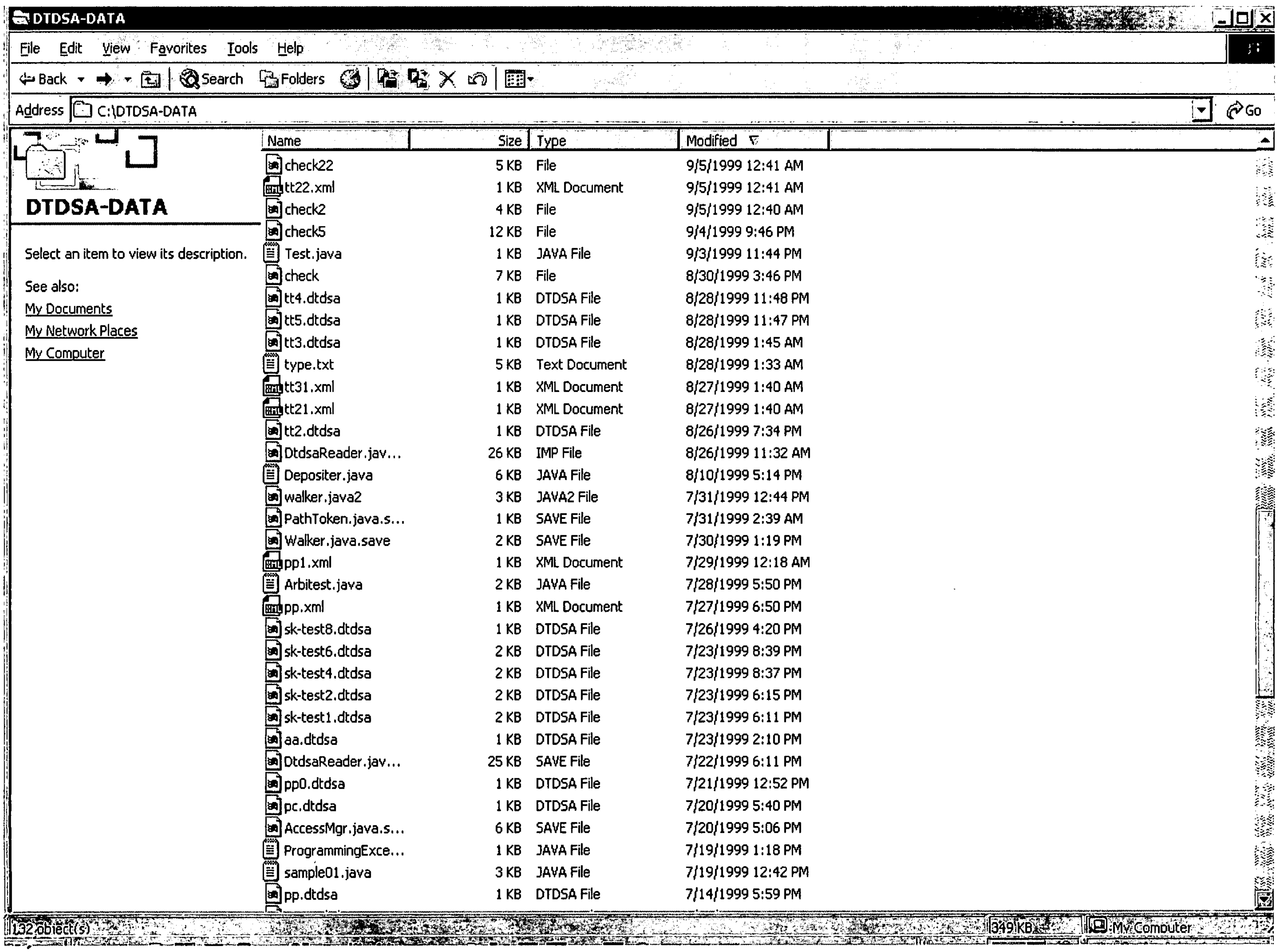
C-1



Name	Size	Type	Modified
test3.dtdsa	1 KB	DTDSA File	7/14/1999 3:02 PM
dtdsa.java.save	2 KB	SAVE File	7/11/1999 2:43 AM
rules.txt	3 KB	Text Document	7/10/1999 12:52 PM
TokenReader.java	2 KB	JAVA File	7/10/1999 12:47 PM
a3.dtdsa	1 KB	DTDSA File	7/9/1999 4:38 PM
a6.dtdsa	1 KB	DTDSA File	7/6/1999 7:23 PM
a5.dtdsa	1 KB	DTDSA File	7/6/1999 7:06 PM
a4.dtdsa	1 KB	DTDSA File	7/6/1999 4:52 PM
a1.dtdsa	1 KB	DTDSA File	6/27/1999 3:48 AM
Node.java.save	4 KB	SAVE File	6/25/1999 6:30 PM
a2.dtdsa	1 KB	DTDSA File	6/25/1999 5:38 PM
NodeList.java.save	1 KB	SAVE File	6/24/1999 6:15 PM
po5.dtdsa	1 KB	DTDSA File	6/21/1999 6:05 PM
po4.dtdsa	1 KB	DTDSA File	6/21/1999 5:11 PM
po3.dtdsa	1 KB	DTDSA File	6/21/1999 5:09 PM
po2.dtdsa	1 KB	DTDSA File	6/21/1999 5:06 PM
po6.dtdsa	1 KB	DTDSA File	6/21/1999 5:06 PM
po1.dtdsa	1 KB	DTDSA File	6/21/1999 2:02 AM
po.dtdsa	1 KB	DTDSA File	6/20/1999 10:56 PM
tt.dtdsa	1 KB	DTDSA File	6/19/1999 4:37 PM
T9.dtdsa	1 KB	DTDSA File	6/19/1999 3:01 AM
T8.dtdsa	1 KB	DTDSA File	6/19/1999 2:59 AM
T7.dtdsa	1 KB	DTDSA File	6/19/1999 2:53 AM
T6.dtdsa	1 KB	DTDSA File	6/19/1999 2:38 AM
sample.txt	1 KB	Text Document	6/17/1999 6:40 PM
TokenReader.jav...	4 KB	SAVE File	6/17/1999 6:35 PM
T5.dtdsa	1 KB	DTDSA File	6/17/1999 6:20 PM
T3.dtdsa	1 KB	DTDSA File	6/17/1999 4:50 PM
D3.dtdsa	1 KB	DTDSA File	6/17/1999 3:08 PM
D2.dtdsa	1 KB	DTDSA File	6/17/1999 3:08 PM
D1.dtdsa	1 KB	DTDSA File	6/17/1999 3:06 PM
T1.dtdsa	1 KB	DTDSA File	6/17/1999 12:53 PM
T4.dtdsa	1 KB	DTDSA File	6/8/1999 5:01 PM
T2.dtdsa	1 KB	DTDSA File	6/8/1999 4:57 PM

132 object(s) 349 KB My Computer

C-2



DTDSA-DATA

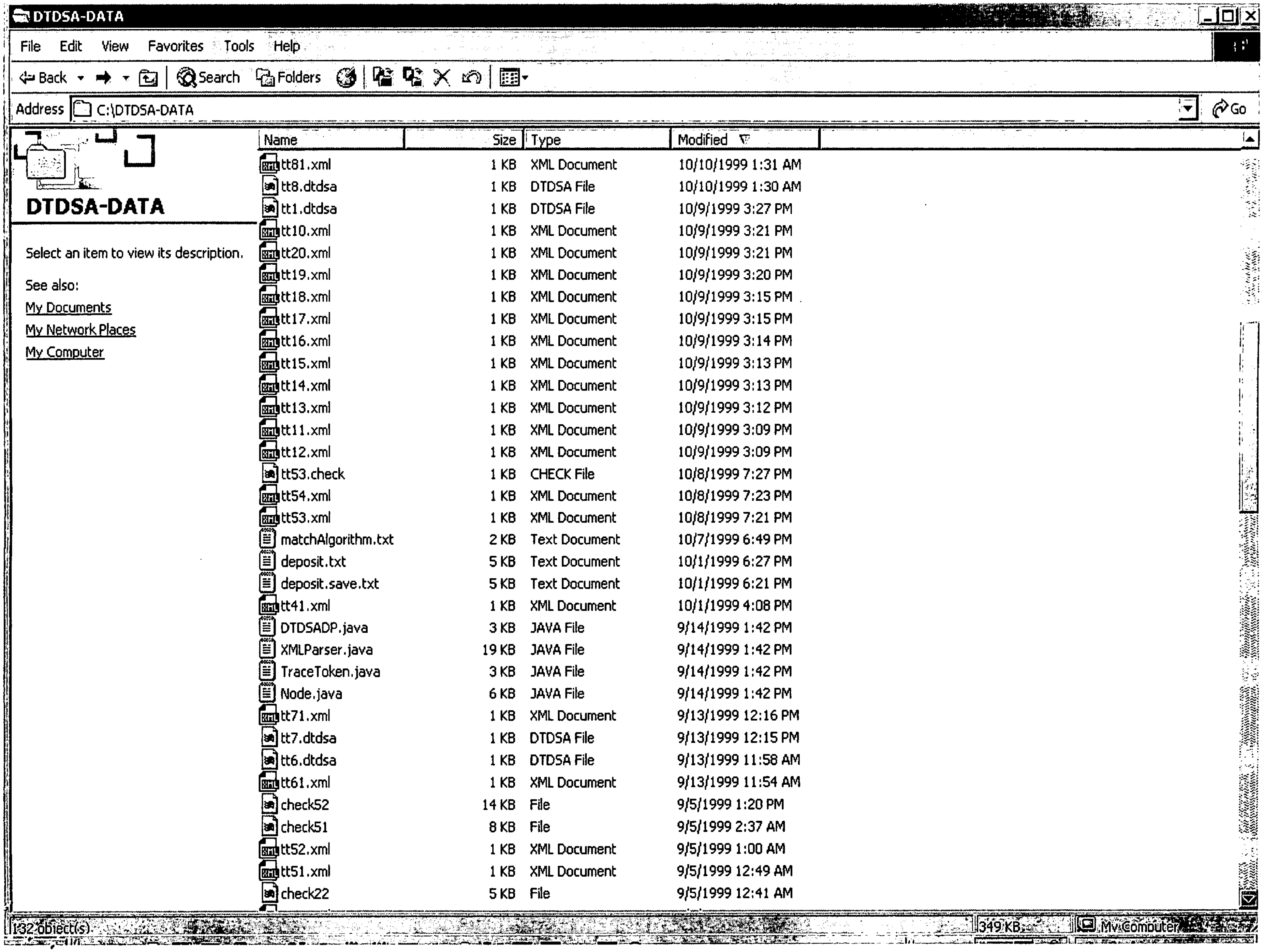
Select an item to view its description.

See also:
My Documents
My Network Places
My Computer

Name	Size	Type	Modified
check22	5 KB	File	9/5/1999 12:41 AM
tt22.xml	1 KB	XML Document	9/5/1999 12:41 AM
check2	4 KB	File	9/5/1999 12:40 AM
check5	12 KB	File	9/4/1999 9:46 PM
Test.java	1 KB	JAVA File	9/3/1999 11:44 PM
check	7 KB	File	8/30/1999 3:46 PM
tt4.dtdsa	1 KB	DTDSA File	8/28/1999 11:48 PM
tt5.dtdsa	1 KB	DTDSA File	8/28/1999 11:47 PM
tt3.dtdsa	1 KB	DTDSA File	8/28/1999 1:45 AM
type.txt	5 KB	Text Document	8/28/1999 1:33 AM
tt31.xml	1 KB	XML Document	8/27/1999 1:40 AM
tt21.xml	1 KB	XML Document	8/27/1999 1:40 AM
tt2.dtdsa	1 KB	DTDSA File	8/26/1999 7:34 PM
DtdsaReader.jav...	26 KB	IMP File	8/26/1999 11:32 AM
Depositer.java	6 KB	JAVA File	8/10/1999 5:14 PM
walker.java2	3 KB	JAVA2 File	7/31/1999 12:44 PM
PathToken.java.s...	1 KB	SAVE File	7/31/1999 2:39 AM
Walker.java.save	2 KB	SAVE File	7/30/1999 1:19 PM
pp1.xml	1 KB	XML Document	7/29/1999 12:18 AM
Arbitest.java	2 KB	JAVA File	7/28/1999 5:50 PM
pp.xml	1 KB	XML Document	7/27/1999 6:50 PM
sk-test8.dtdsa	1 KB	DTDSA File	7/26/1999 4:20 PM
sk-test6.dtdsa	2 KB	DTDSA File	7/23/1999 8:39 PM
sk-test4.dtdsa	2 KB	DTDSA File	7/23/1999 8:37 PM
sk-test2.dtdsa	2 KB	DTDSA File	7/23/1999 6:15 PM
sk-test1.dtdsa	2 KB	DTDSA File	7/23/1999 6:11 PM
aa.dtdsa	1 KB	DTDSA File	7/23/1999 2:10 PM
DtdsaReader.jav...	25 KB	SAVE File	7/22/1999 6:11 PM
pp0.dtdsa	1 KB	DTDSA File	7/21/1999 12:52 PM
pc.dtdsa	1 KB	DTDSA File	7/20/1999 5:40 PM
AccessMgr.java.s...	6 KB	SAVE File	7/20/1999 5:06 PM
ProgrammingExce...	1 KB	JAVA File	7/19/1999 1:18 PM
sample01.java	3 KB	JAVA File	7/19/1999 12:42 PM
pp.dtdsa	1 KB	DTDSA File	7/14/1999 5:59 PM

132 object(s) | 349 KB | My Computer

C-3



DTDSA-DATA

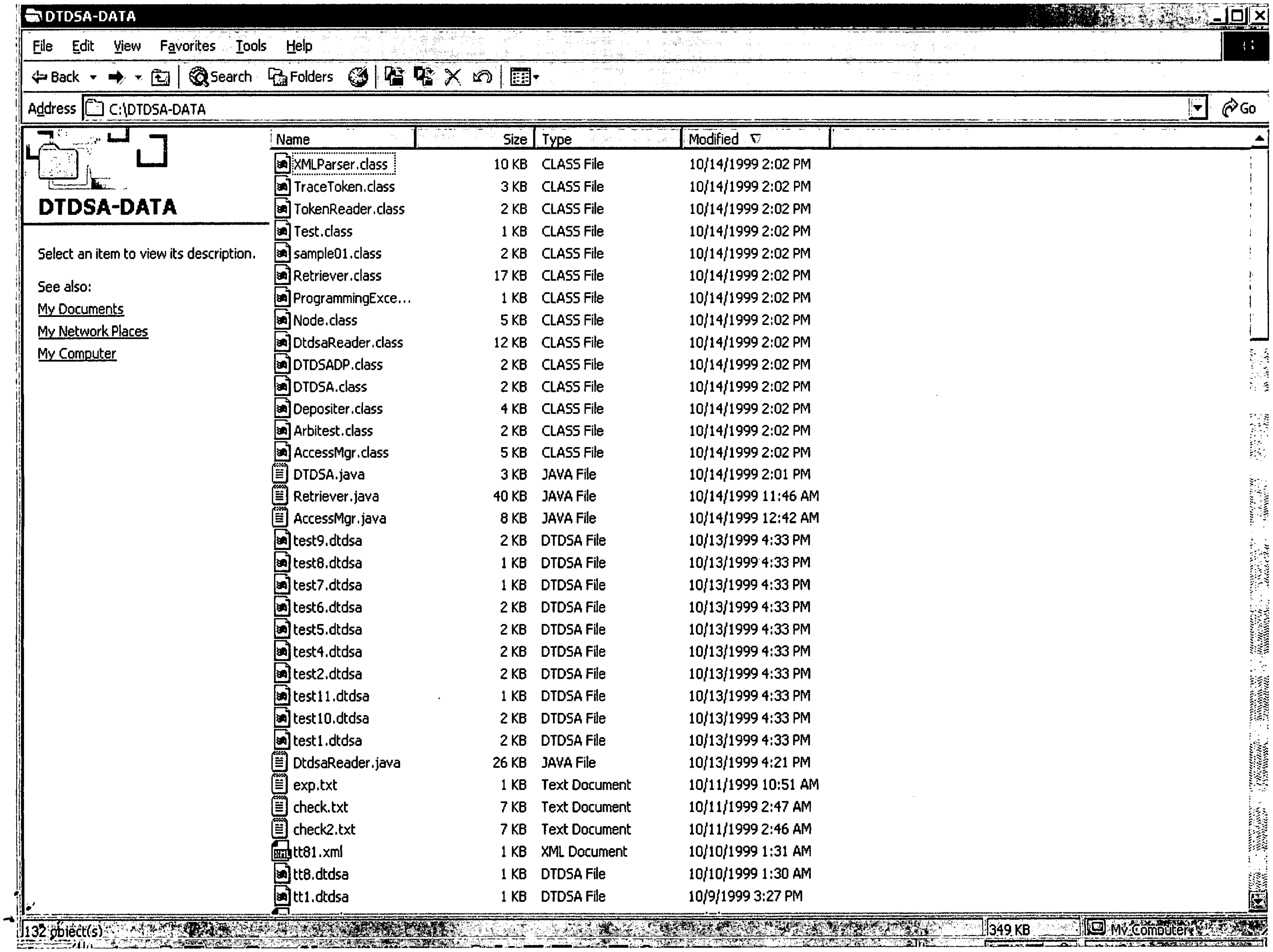
Select an item to view its description.

See also:
My Documents
My Network Places
My Computer

Name	Size	Type	Modified
tt81.xml	1 KB	XML Document	10/10/1999 1:31 AM
tt8.dtdsa	1 KB	DTDSA File	10/10/1999 1:30 AM
tt1.dtdsa	1 KB	DTDSA File	10/9/1999 3:27 PM
tt10.xml	1 KB	XML Document	10/9/1999 3:21 PM
tt20.xml	1 KB	XML Document	10/9/1999 3:21 PM
tt19.xml	1 KB	XML Document	10/9/1999 3:20 PM
tt18.xml	1 KB	XML Document	10/9/1999 3:15 PM
tt17.xml	1 KB	XML Document	10/9/1999 3:15 PM
tt16.xml	1 KB	XML Document	10/9/1999 3:14 PM
tt15.xml	1 KB	XML Document	10/9/1999 3:13 PM
tt14.xml	1 KB	XML Document	10/9/1999 3:13 PM
tt13.xml	1 KB	XML Document	10/9/1999 3:12 PM
tt11.xml	1 KB	XML Document	10/9/1999 3:09 PM
tt12.xml	1 KB	XML Document	10/9/1999 3:09 PM
tt53.check	1 KB	CHECK File	10/8/1999 7:27 PM
tt54.xml	1 KB	XML Document	10/8/1999 7:23 PM
tt53.xml	1 KB	XML Document	10/8/1999 7:21 PM
matchAlgorithm.txt	2 KB	Text Document	10/7/1999 6:49 PM
deposit.txt	5 KB	Text Document	10/1/1999 6:27 PM
deposit.save.txt	5 KB	Text Document	10/1/1999 6:21 PM
tt41.xml	1 KB	XML Document	10/1/1999 4:08 PM
DTDSADP.java	3 KB	JAVA File	9/14/1999 1:42 PM
XMLParser.java	19 KB	JAVA File	9/14/1999 1:42 PM
TraceToken.java	3 KB	JAVA File	9/14/1999 1:42 PM
Node.java	6 KB	JAVA File	9/14/1999 1:42 PM
tt71.xml	1 KB	XML Document	9/13/1999 12:16 PM
tt7.dtdsa	1 KB	DTDSA File	9/13/1999 12:15 PM
tt6.dtdsa	1 KB	DTDSA File	9/13/1999 11:58 AM
tt61.xml	1 KB	XML Document	9/13/1999 11:54 AM
check52	14 KB	File	9/5/1999 1:20 PM
check51	8 KB	File	9/5/1999 2:37 AM
tt52.xml	1 KB	XML Document	9/5/1999 1:00 AM
tt51.xml	1 KB	XML Document	9/5/1999 12:49 AM
check22	5 KB	File	9/5/1999 12:41 AM

132 object(s) 349 KB My Computer

C-4



DTDSA-DATA

File Edit View Favorites Tools Help

Back Search Folders

Address C:\DTDSA-DATA

DTDSA-DATA

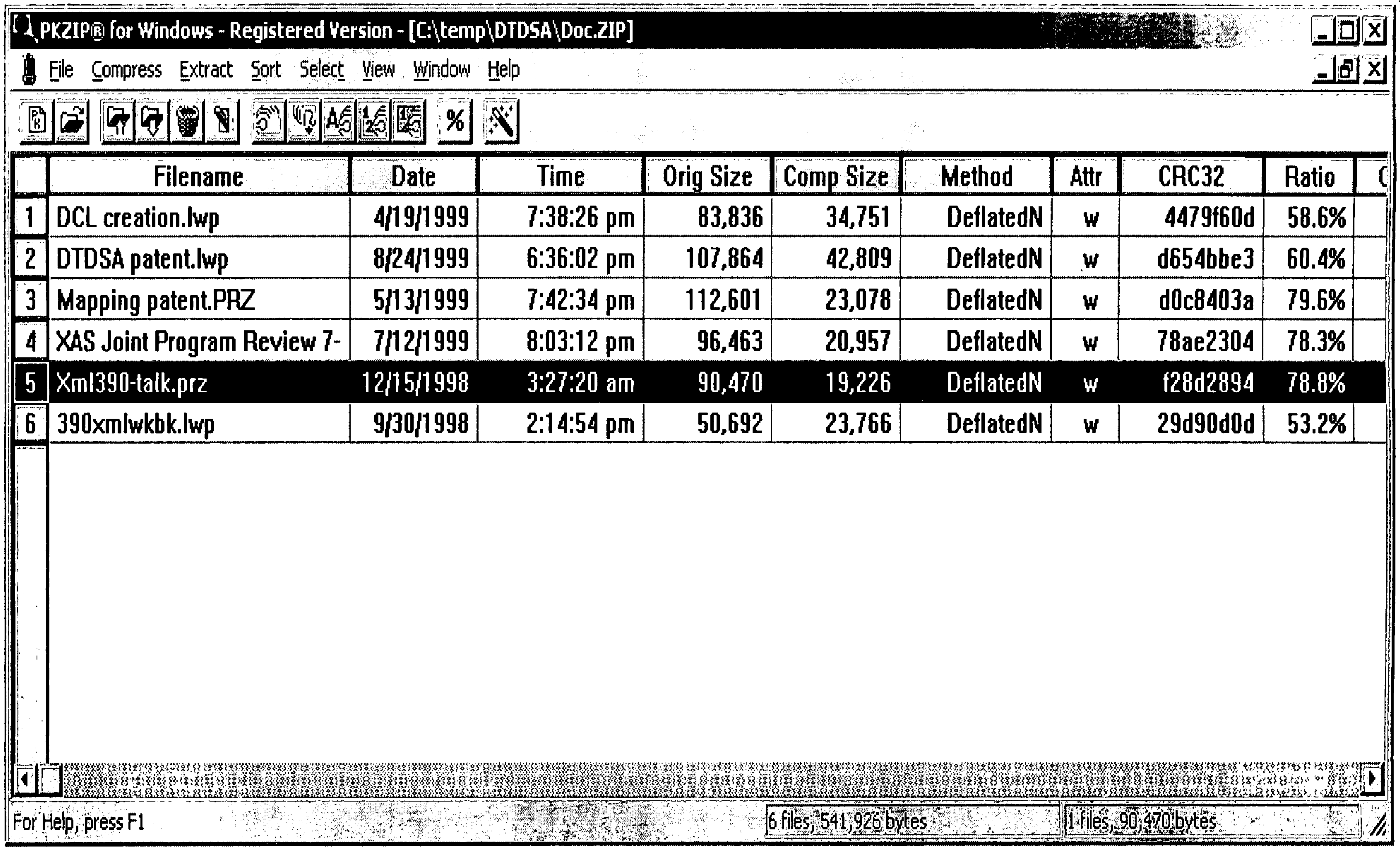
Select an item to view its description.

See also:
My Documents
My Network Places
My Computer

Name	Size	Type	Modified
XMLParser.class	10 KB	CLASS File	10/14/1999 2:02 PM
TraceToken.class	3 KB	CLASS File	10/14/1999 2:02 PM
TokenReader.class	2 KB	CLASS File	10/14/1999 2:02 PM
Test.class	1 KB	CLASS File	10/14/1999 2:02 PM
sample01.class	2 KB	CLASS File	10/14/1999 2:02 PM
Retriever.class	17 KB	CLASS File	10/14/1999 2:02 PM
ProgrammingExce...	1 KB	CLASS File	10/14/1999 2:02 PM
Node.class	5 KB	CLASS File	10/14/1999 2:02 PM
DtdsaReader.class	12 KB	CLASS File	10/14/1999 2:02 PM
DTDSADP.class	2 KB	CLASS File	10/14/1999 2:02 PM
DTDSA.class	2 KB	CLASS File	10/14/1999 2:02 PM
Depositer.class	4 KB	CLASS File	10/14/1999 2:02 PM
Arbitest.class	2 KB	CLASS File	10/14/1999 2:02 PM
AccessMgr.class	5 KB	CLASS File	10/14/1999 2:02 PM
DTDSA.java	3 KB	JAVA File	10/14/1999 2:01 PM
Retriever.java	40 KB	JAVA File	10/14/1999 11:46 AM
AccessMgr.java	8 KB	JAVA File	10/14/1999 12:42 AM
test9.dtdsa	2 KB	DTDSA File	10/13/1999 4:33 PM
test8.dtdsa	1 KB	DTDSA File	10/13/1999 4:33 PM
test7.dtdsa	1 KB	DTDSA File	10/13/1999 4:33 PM
test6.dtdsa	2 KB	DTDSA File	10/13/1999 4:33 PM
test5.dtdsa	2 KB	DTDSA File	10/13/1999 4:33 PM
test4.dtdsa	2 KB	DTDSA File	10/13/1999 4:33 PM
test2.dtdsa	2 KB	DTDSA File	10/13/1999 4:33 PM
test11.dtdsa	1 KB	DTDSA File	10/13/1999 4:33 PM
test10.dtdsa	2 KB	DTDSA File	10/13/1999 4:33 PM
test1.dtdsa	2 KB	DTDSA File	10/13/1999 4:33 PM
DtdsaReader.java	26 KB	JAVA File	10/13/1999 4:21 PM
exp.txt	1 KB	Text Document	10/11/1999 10:51 AM
check.txt	7 KB	Text Document	10/11/1999 2:47 AM
check2.txt	7 KB	Text Document	10/11/1999 2:46 AM
tt81.xml	1 KB	XML Document	10/10/1999 1:31 AM
tt8.dtdsa	1 KB	DTDSA File	10/10/1999 1:30 AM
tt1.dtdsa	1 KB	DTDSA File	10/9/1999 3:27 PM

132 object(s) 349 KB My Computer

Exhibit D



PKZIP® for Windows - Registered Version - [C:\temp\DTDSA\Doc.ZIP]

	Filename	Date	Time	Orig Size	Comp Size	Method	Attr	CRC32	Ratio	C
1	DCL creation.lwp	4/19/1999	7:38:26 pm	83,836	34,751	DeflatedN	w	4479f60d	58.6%	
2	DTDSA patent.lwp	8/24/1999	6:36:02 pm	107,864	42,809	DeflatedN	w	d654bbe3	60.4%	
3	Mapping patent.PRZ	5/13/1999	7:42:34 pm	112,601	23,078	DeflatedN	w	d0c8403a	79.6%	
4	XAS Joint Program Review 7-	7/12/1999	8:03:12 pm	96,463	20,957	DeflatedN	w	78ae2304	78.3%	
5	Xml390-talk.prz	12/15/1998	3:27:20 am	90,470	19,226	DeflatedN	w	f28d2894	78.8%	
6	390xmlwkbk.lwp	9/30/1998	2:14:54 pm	50,692	23,766	DeflatedN	w	29d90d0d	53.2%	

For Help, press F1 | 6 files, 541,926 bytes | 1 files, 90,470 bytes

OIPE JC151 JUN 1 7 2004 PATENT & TRADEMARK OFFICE

Appendix Z

Anne Barschall

From: "vt" <turau@tuhh.de>
To: "Anne Barschall" <anne.barschall@worldnet.att.net>
Sent: Friday, May 14, 2004 2:55 AM
Subject: Re: Information request

Hi,

well the ideas for DB2XML started in second half of 1998, in september 1998 I started with the implementation and documentation. The first public release was made on March, 1, 1999 (as documented on the web page http://www-1.informatik.fh-wiesbaden.de/~turau/DB2XML/releaseNotes.html,
which is mirrored on several sites over the world). The documentation of release 1.0 contained most of the stuff contained in the paper. After the release of Version 1.0 on May, 20 1999 I started to write the paper you mentioned, I finished it during the summer and submitted it to a conference. It was rejected. In parallel I published it as a technical report through the normal university publishing process (i.e. I got a report number for it, TR-99-103). This is a normal process to document my research for the university review process. That was in October 1999, I cannot state the exact date. I gues it was in the first or second week, since the term stared in the third week and I usually finish such work before the lectures start. Anway the main ideas were born in 1998 and documented in early 1999.

Please give some details about your that patent application.

volker turau

Volker Turau
Prof. Dr. rer. nat.
Technische Universität Hamburg-Harburg

Schwarzenbergstr. 95
D-21073 Hamburg
Tel.: (+4940) 42878-3530

Z-1

Fax: (+4940) 42878-2581
E-Mail: turau@tuhh.de

On Thu, 13 May 2004, Anne Barschall wrote:

> Dear Dr. Turau,
>
> In an e-mail to Shirelle Green, dated November 5, 2003, you stated that the
> date of publication of your paper "Making legacy data accessible for XML
> applications" was published in October 1999 as a technical note.
>
> Do you happen to recall what day in October it was published? Also, what
> does it mean to be published as a "technical note?" What is involved with
> that?
>
> Thank you for your help.
>
> Very truly yours,
>
> Anne Barschall
>
>
>

Release Notes

October, 1 2001 Version 1.4

- Bug-Fixes
- Ant Makefile
- Interface for JAXP-conform XML parsers and XSLT processors
- Better documentation (for servlet application)

January, 31 2000 Version 1.3

- Preliminary support for JDBC 2.0 types (Clob, Blob)
- XSLLotusParser now works with xml4j (2_0_15) and lotusxsl (0_19_2)
- Re-implementation of some core classes
- Improved command line tool
- More properties
- Bug fixes

August, 31 1999 Version 1.2

- Better support for XSLT stylesheet parsing
- Integrated XSLT stylesheet processing based on the lotus parser
- Better support for converting complete databases
- Changed handling of formatting of currencies and numbers
- Bug fixes

June, 21 1999 Version 1.1

- DOM interface
- Bug fixes
- Tutorial on how to use DB2XML
- New example

May, 20 1999 Version 1.0

- Source code release
- Configurable representation for the types Time, Date and Timestamp
- Locale specific representation of currencies
- Bug fixes

May, 4 1999 Version 0.9

- Primary key attribute
- Support for different character encodings

Z73

- Loading and storing of properties
- Command line tool
- Code redesign

April, 15 1999 Version 0.8

- Support for multiple queries and full databases
- User defined names for element types
- Hierarchical names for element types
- New attributes

March, 24 1999 Version 0.7

- Introduced attribute NAME.
- Servlet version
- Support for stylesheets
- New GUI for main panel
- Reorganization of this document
- minor bug fixes

March, 10 1999 Version 0.6

- External DTDs
- External references for binaries
- Username and password handling
- Default values for various properties can now be set in the file `db2xml.properties`
- Changed type of attribute TYPE. It is now an enumeration of all local type names used in the current query as opposed to **all** legal local type names. In case a driver cannot determine these type names, the type of attribute TYPE is `CDATA`.

March, 1 1999 Version 0.5

- Public release.

Contact

Technical questions, comments, and bug reports to turau@informatik.fh-wiesbaden.de.

Please include information about your environment:

Z-9

name and version of database, name and version of driver, operating system and version of DB2XML.

Author

Volker Turau Authors Home Page

7-5